Data mining: a database perspective.

M. S. Sousa, M. L. Q. Mattoso & N. F. F. Ebecken
COPPE, Federal University of Rio de Janeiro
P.O. Box 68511, Rio de Janeiro, RJ, Brazil, 21945-970
EMail: mauros, marta @cos.ufrj.br, & nelson@ntt.ufrj.br

Abstract

Data mining on large databases has been a major concern in research community, due to the difficulty of analyzing huge volumes of data using only traditional OLAP tools. This sort of process implies a lot of computational power, memory and disk I/O, which can only be provided by parallel computers. We present a discussion of how database technology can be integrated to data mining techniques. Finally, we also point out several advantages of addressing data consuming activities through a tight integration of a parallel database server and data mining techniques.

1 Introduction

Data mining techniques have increasingly been studied[7,9,21], especially in their application in real-world databases. One typical problem is that databases tend to be very large, and these techniques often repeatedly scan the entire set. Sampling has been used for a long time, but subtle differences among sets of objects become less evident.

This work provides an overview of some important data mining techniques and their applicability on large databases. We also spot several advantages of using a database management system (DBMS) to manage and process information instead of conventional flat files. This approach has been a major concern of several researches, because it represents a very natural solution since DBMSs have been successfully used in business management and currently may store valuable hidden knowledge.

One requirement of data mining is efficiency and scalability of mining algorithms. It makes the use of parallelism even more relevant to provide a way of processing long running tasks in a timely manner. In this context, parallel database systems come to play an important role, because they can offer, among other advantages, transparent and painless implementation of parallelism to process large data sets. It is important to notice that, when we mention the use of large amounts of information in data mining, we are not referring to usual large DBMSs, which can reach more than one terabyte of data. As data mining methods often repeatedly scan the data set, mining in such a large database is not cited in the literature yet.

The remainder of this work is organized as follows. Section 2 presents some important mining techniques currently implemented in data mining systems. Section 3 describes how these techniques can be applied to both flat files and DBMSs, enforcing advantages of the latter approach, and Section 4 presents important advantages to be analyzed when considering the use of parallel databases to mine knowledge. Section 5 describes a case study and an implementation using a well-known classifier algorithm with a tightly-coupled integration with a database system. Section 6 points out some related data mining systems, whereas Section 7 presents our conclusions and final observations.

2 Data Mining Techniques

Data mining is a step in knowledge discovery in databases (KDD) that searches for a series of hidden patterns in data, often involving a repeated iterative application of particular data mining methods. The goal of the whole KDD process is to make patterns understandable to humans in order to facilitate a better interpretation of the underlying data[11].

We present four classes of data mining techniques typically used in a variety of well-known applications and researches currently cited in the database mining community. They certainly do not represent all mining methods, but are a considerable portion of them when a large amount of data is considered.

Transactions on Information and Communications Technologies vol 19 © 1998 WIT Press, www.witpress.com, ISSN 1743-3517

Classification is a well-known data mining operation and it has been studied in machine learning community for a long time[8,18,19,24]. Its aim is to classify cases into different classes, based on common properties (attributes) among a set of objects in a database. After the construction of the classification model, it is used to predict classes of new cases that are going to be inserted in the database. Adequate applications for classification include medical diagnosis, credit risk assessment, fraud detection and target marketing.

Among classification algorithms, there are two methods that are widely used for data mining purposes: decision trees and neural networks.

2.1.1 Decision Trees

Decision tree methods are a kind of machine learning algorithm that uses a divide-and-conquer approach to classify cases using a tree-based representation[8,24]. They usually use a greedy algorithm that recursively subdivides the training set until reaching a partition that represents cases totally belonging to the same class or until a criteria is reached (pre-pruning). When deciding what attribute is going to be used by each subdivision, a statistical test is adopted as the splitting criteria.

After the growing phase, we may have an overspecialized tree that overfits data, providing more structure than necessary. Pruning comes to play an important role in producing smaller trees with better accuracy when considering new cases[24].

Even after pruning, trees can represent a complex, difficult to understand structure. Rule extraction is the final phase used to get smaller and less complex rules with similar accuracy. A greedy covering algorithm is often used to select a minimal subset of rules that covers the examples[24].

The main problem with trees is that they need extensive data to uncover complex structures. On the other hand, they can be constructed considerably faster than other machine learning algorithms producing results with similar accuracy[21].

There are many implementations using decision tree learning algorithms, such as CART[8], ID3 and its successor C4.5[24], and SPRINT (SLIQ's parallel implementation)[25,19].

Transactions on Information and Communications Technologies vol 19 © 1998 WIT Press, www.witpress.com, ISSN 1743-3517

Methods based on artificial neural networks[7] provide a general and practical method for learning functions, which are represented by continuous attributes, discrete or vectors. One important characteristic of the algorithm is its robustness when dealing with errors in the training set. Basically, neural networks have been used to interpret visual scenes, voice recognition, and they are not only used for classification (e.g. neural networks are widely used for prediction purposes).

Neural networks typically require more time to finish than, say, decision trees algorithms[21]. Training times vary depending on the number of training cases, the number of weights in the network, and the settings of many learning algorithm parameters. Finally the ability to understand learned target function is not important. Learned neural networks are less easily communicated to humans than learned rules.

2.2 Association Rules

Mining association rules[2,6] is particularly important when trying to find relevant associations among items in a given customer transaction. An example of the output of such mining is the statement that 80% of transactions that purchase diapers and milk also purchase milk bottles. The number 80% is the rule confidence factor.

A formal depiction of the problem is presented in Agrawal et al[2]. Let $I = \{i_1, i_2, i_3, .., i_m\}$ be a set of binary attributes called items and let T be a database of transactions. Each transaction t is represented by a binary vector, with $t[k] = 1$ if t bought the item i_k, and $t[k] = 0$ otherwise. It is interesting to notice that we are not, at this moment, worried about quantities, that is, how many items were bought in each transaction.

Now let X be a set of items, such that X is a subset of I. We may say that a transaction t satisfies X if, for all items in X, $t[k] = 1$. An association rule is an implication of the form $X \Rightarrow Y$, where X is a subset of I, Y is a subset of I, and $X \bigcap Y = \emptyset$. If $c\%$ of transactions in D that contain X also contain Y, thus $c\%$ represents the rule confidence factor. The rule $X \Rightarrow Y$ has a support $s\%$ if $s\%$ of transactions in D contain $X \Rightarrow Y$.

The problem of mining association rules can be decomposed into

Transactions on Information and Communications Technologies vol 19 © 1998 WIT Press, www.witpress.com, ISSN 1743-3517

- Generate all combinations of items that a have a transaction support above an informed minimum threshold (large item sets). This phase is considered the most time consuming task of the algorithm.

- For all large item sets, generate association rules for the transaction database, which is done in a straightforward manner.

2.3 Clustering

Clustering algorithms[3], also called unsupervised classification, is the process of grouping physical or abstract objects into classes of similar objects. Clustering analysis helps to construct meaningful partitionings of a large set of objects based on a *divide and conquer* methodology, which decomposes a large-scale system into smaller components to simplify design and implementation.

2.4 Sequential Patterns

The discovery of sequential patterns[28] has been motivated by applications in retailing industry, including attached mailing and add-on sales, and in the medical domain, for example.

The input data is typically a list of sequential transactions and there is often a transaction-time associated with each of them. The primary goal is to find all sequential patterns with a pre-defined minimum support, that is, the percentage of data sequences that contains the pattern is greater than a given threshold.

Thus, sequential pattern algorithms are useful for discovering trends in data, such as:

The number of sports magazines sold to customers with credit between 20,000 and 30,000 living in city 2 is increasing.

3 Using Large Databases

One important characteristic of the data mining process is that it can use data that has been gathered during many years and transform it in valuable knowledge. The larger the database, the better and the more accurate the knowledge. An important observation is that, when we refer to the use of large amounts of information in data

mining we are not referring to usual large DBMSs, which can reach terabytes. As data mining methods often repeatedly scan the data set, mining in such a large database is far from being a reality.

Data mining applications often work with two kinds of data sources, and each one has its own advantages: flat files or a database management system.

3.1 Flat Files

Flat files have been largely used as the data source of information used by data mining algorithms. Even when operational data is stored in a DBMS, many systems extract a portion of them to be used during the mining process, working only with data that fits in main memory, which limits the total amount of information to be used. Implementations using flat files offer some advantages related to performance results.

3.2 Data Mining and DBMSs

Database technology has been successfully used in traditional business data processing. Companies have been gathering a large amount of data, using a DBMS system to manage it. Therefore, it is desirable that we have an easy and painless use of database technology within other areas, such as data mining.

DBMS technology offers many features that make it valuable when implementing data mining applications. For example, it is possible to work with data sets that are considerably larger than main memory, since the database itself is responsible for handling information, paging and swapping when necessary. Besides, a simplified data management and a closer integration to other systems are available (e.g. data may be updated or managed as a part of a larger operational process). Moreover, as emerging object-relational databases are providing the ability to handle image, video and voice, there is a potential area to exploit mining of complex data types. Finally, after rules are discovered, we can use ad-hoc and OLAP queries to validate discovered patterns in an easy way. We must not forget that information used during mining processing is often confidential. Thus, DBMSs can also be used as a means of providing data security, which is widely implemented in commercial databases, avoiding the need of using encryption algorithms to process information.

Transactions on Information and Communications Technologies vol 19 © 1998 WIT Press, www.witpress.com, ISSN 1743-3517

Even when using databases, most of the current data mining applications have a loose connection with them. They treat database simply as a container from which data is extracted directly to the main memory of the computer responsible for running the data mining algorithm, just before the main execution begins. This approach limits the amount of data that can be used, forcing applications to filter information, and use only a part of it to discover patterns. Alternatively, some applications dynamically perform queries to the database, but work in a client / server architecture. It means that, depending on the amount of data being transferred, unnecessary network traffic is generated. Moreover, they are often written in programming languages that do not have any integration with the database system.

3.2.2 Tightly-coupled [14]

The idea of executing user-defined computation within databases, thus avoiding unnecessary network traffic, has been a major concern in many DBMS systems. One example of this sort of implementation is the stored procedure in commercial databases. For instance, Oracle provides the possibility of implementing procedural code in PL/SQL and storing it within the data dictionary, in a pre-compiled form (PL/SQL is an interpreted language).

Some authors have developed a methodology for tightly-coupled integration of data mining applications with database systems[5], selectively pushing parts of the application program into them. They have performed some practical experiments with a relational database system and propose that, whenever possible, computation should be included within SQL statements.

4 Data Mining and Parallel DBMSs

Given that it is desirable that a data mining process handles a large volume of data, parallel algorithms are needed to provide scalability, in order to end the process in a timely manner. Classification and discovery of association rules are two data mining techniques that have been extensively studied for parallelization purposes[4,6,17,18,26].

It can be explained by several reasons, such as the importance of these methods for commercial use, and their scalability.

There are a variety of data mining algorithms constructed to run in parallel, taking advantage of parallel architectures using specific programming routines. Alternatively, parallel database systems can provide parallelization in a transparent, painless manner to the application. There are many advantages of using parallel DBMSs:

- *The implementation becomes simpler.* There is no need to use parallel routines such as MPI libraries. The parallel DBMS is responsible itself for parallelizing queries that are issued against it. We have to structure our queries so that they can fully exploit parallelism offered by a particular DBMS.

- *DBMS can choose between serial and parallel execution plans transparently to application.* Depending on the data mining algorithm and technique used, it is clear that parallelism is more useful in the beginning and intermediate phases of the process. In the construction of decision trees, for example, the training set is recursively partitioned and processed. Thus, at some point of the processing, parallelism may not be useful because the communication overhead between coordinator and slave processes dominates the processing time, because there are not many training examples being searched. Based on statistics about data, the DBMS can evaluate the cost of parallel and serial queries and decide which one is more advantageous, depending on available machine resources.

- *Opportunity for database fragmentation.*
Database fragmentation[20,22] provides many advantages related to the reduced number of page accesses necessary for reading data. Irrelevant data could be just ignored depending on the filter specified by SQL statements. Furthermore, some DBMSs can process each partition in parallel, using different execution plans when applicable[23]. In general, DBMS systems provide automatic management of such functionality.

However, some drawbacks have to be considered, such as:

- *Less control of how parallelization will occur.* Although there is the possibility of some customizations, such as setting the

nodes that will participate in the process, the DBMS itself is in charge for parallelization.

- *The algorithm may become dependent on some characteristic of the parallel DBMS used.* In a parallel database system, we have some chance to direct the optimizer to use a specific execution plan, but our algorithm may be dependent on some characteristic of this DBMS.

- *Overhead of the database system kernel.* A kernel of a database system is designed to handle a large set of operations, such as OLTP transactions, OLAP queries, etc. Although it is possible to minimize the overhead and customize the environment to take the best available advantages of the corresponding architecture, there will always exist some functionality implemented that is not applicable to the data mining algorithm, which can degrade performance when compared to access to flat files.

5 A Classification Case Study

To study and identify problems related to database mining on a large amount of data, we have implemented a classification algorithm (decision tree) with a tightly-coupled integration to a parallel database server. We have constructed a set of stored procedures that are stored within Oracle Server data dictionary in a pre-compiled form. We have been using a multiprocessor IBM RS/6000 SP2 system with Oracle Parallel Server 7.3 installed[14,23].

We have decided to base our implementation on decision tree algorithms because they are well-suited classifiers for data mining problems, producing similar accuracy when compared to other methods for classification. Besides, they represent one of the most widely used and practical methods for inductive inference, and they can be built considerably faster than other machine learning methods. It is an important characteristic when focusing on handling a large amount of data. Moreover, decision trees can be easily constructed from and transformed into SQL statements[1], which can be used to query a parallel database system.

Another important consideration about our implementation is that it does not have any memory dependency, that is, it will wok no matter how large the data set is. If data does not fit in memory,

the DBMS itself is responsible for the paging and swapping when necessary. We do not propose any new data mining algorithm, but a different approach of implementing a data mining system. Our work is based on C4.5[24] and SPRINT[25], adapting them to use a database system instead of flat files. A more detailed discussion of database technology features appears in [26], while the description of the implemented decision tree algorithm in Oracle Parallel Server may be found in [27].

5.1 Problem Statement

The data set used during the classification process is based on a life insurance system, which controls information about customers, insurance contracts and components of insurance tariffs. People in management positions use this database in a data-warehousing environment in order to help decision making. Information about customers and their dependents includes year of birth, job, sex, marital status, and household income data. Besides, the system holds data about contracts, such as the year when contract begins and ends, modus of payment, name of the agent responsible for the contract, and type of insurance. Finally, the database keeps information about price of each tariff that is associated to an insurance contract.

Although a series of normalized tables were available, we have constructed a single relation because of performance problems related to join operations. Oracle offers a variety of methods to perform joins in parallel, namely nested loops, hash joins and star joins. However, even when performed in parallel, joins represent a time consuming task. This single relation is a relatively small one that fits in memory, with about 50Mb, 200.000 tuples and 70 attributes. For data mining purposes, this relation represents a considerable large one when compared to others that appear in current literature.

We handle attributes with different characteristics, namely: discrete attributes with very few distinct values, discrete attributes with a reasonable number of distinct values, continuous attributes with only some distinct attributes and continuous attributes with a lot of distinct values.

5.2 Typical Queries in Rule Induction Algorithms

Most of queries issued in decision tree algorithms, or in rule induction ones in general, deals with two kinds of operations: grouping (GROUP BY) and sorts (ORDER BY), when working with continuous attributes[12]. These kinds of queries represent the time consuming tasks obtained during the implemented algorithm.

Select marital_status, class, count(*)
From mine_relation
Group By marital_status, class

Parallel database servers can easily process this sort of query, and it is possible to take advantage of parallel machine resources. In Massively Parallel Processing architectures (MPP), for example, each node can independently compute counts on the records stored locally. In the next phase, a coordinator process can consolidate these counts, and then return the result back to the user[20].

5.3 Tightly-coupled Implementation

Due to the importance of database technology, we have focused on implementing most of classification algorithm within database. This approach offers a tightly-coupled integration with the database system and its query language and also avoids network traffic between a client and the server. All processing is done in the server and intermediate and final results are also stored within it in the form of tables. The client only issues a command (a stored procedure call) and the DBMS is responsible for extracting results. Moreover, it decreases the number of calls to the database, makes our code considerably simpler, and pushes memory consideration problems into the database system.

Our implementation involves the use of embedded routines written PL/SQL (Oracle native procedural language), which are, whenever possible, called within SQL statements.

5.3.1 Tree Growing Phase

The following query presents an example of SQL statement used during the growing phase of a decision tree, and it is used to calculate the best splitting point of an internal node using gini index statistical test[8,25]. The inner SQL statement groups the training set based on

attribute values. The GROUP BY clause is used instead of a simple ORDER BY to count the number of occurrences for each pair attribute/class, since we cannot split cases having the same attribute value. Hence, the splitting criteria is computed faster than before, using only one database call per attribute in each phase. When the attribute has too many distinct values, a discretization would be advisable.

Select min(giniSplit(salary, class, num))
From (Select salary, class, count(*) num
From mine_relation
Group By salary, class)

5.3.2 Tree Pruning Phase

During the pruning step, a user-defined function is created based on the original tree. It is used to classify test cases in parallel.

Select class, Classify(marital_status, year_contract_end, salary)
From mine_relation_test

We have randomly divided the data set into two parts: one for the tree growing phase, and the other for tree pruning. This task was carefully performed to generate relations with the same class distribution. Test data is represented by mine_relation_test, which is used to predict error rates associated with each tree node.

In the above SQL statement, the first column returns actual case classification, while Classify returns the classification generated by the algorithm. Function Classify is used to find out the number of hits and misses during classification, which will guide tree pruning.

5.3.3 Rule Extraction Phase

Once the classification model is constructed, a set of rules can be derived reading the tree, from top to bottom, until reaching each leaf node. The n rules originally defined (where n represents the number of leaf nodes) are generalized, and some of them may be simplified or even eliminated. Rule generalization process is a time consuming task, since we examine many possibilities in order to eliminate irrelevant expressions in rules, producing a simpler subset. This process issues many SQL statements, in which WHERE conditions represent rules discovered in previous phases. These statements, each one

processed in parallel, return the number of false positives and false negatives examples that are going to be used in conjunction with a well-known method called Minimum Description Length (MDL), as it is done in C4.5[24].

As in the pruning case, this rule set is transformed into a PL/SQL function, which can be called from within a SQL statement, enabling the classification of new cases in parallel. This function receives the attributes used during the data mining algorithm and returns the predicted classification.

5.4 Performance Considerations

Depending on the database size and the complexity of applications, the data mining process may represent a long running task. In our case study, a tree with over 38.000 nodes was generated, what shows that it is not a trivial problem to be solved. Most of parallelization occurs when handling decision nodes that are on the top of tree, so that more data is going to be analyzed. For these phases, we noticed a significant performance speedup when additional processors are used.

Although ad-hoc file processing algorithms often outperform the ones based on DBMSs, there are many advantages of implementing the latter approach. In our study, code simplicity proved to be one of the most valuable advantages to be considered.

Performance improvements could be achieved through data fragmentation techniques. One characteristic of SQL that can degrade the performance of a Rule Induction algorithm is that its structure requires a separate query to construct the histogram for each attribute. Thus, it is not possible to perform a single SQL statement that can bring us the result reading data just once. Fragmentation techniques tend to eliminate irrelevant information during query execution, minimizing such problem. Vertical fragmentation can be used to break a big relation into smaller pieces, whereas horizontal partitioning functions can be carefully defined in order to diminish the amount of information that needs to be scanned in each decision node.

All implemented modules do not rely on any specific Oracle Server feature. They use general functionality, which is currently available in most of commercial parallel database systems. Moreover, this implementation can be easily used by another application, since the constructed stored procedures / functions are just ready to be used in a parameterized, flexible form.

6 Overview of Data Mining Systems

In this section we present some currently implemented data mining systems and some other interesting works in the area. They all have in common the fact that they were implemented taking large amounts of data into account.

6.1 Quest

The Quest data mining system[4] has been addressing the development of fast, scalable algorithms. Implemented algorithms include association rules, generalization, sequential patterns, time-series clustering and classification. Quest algorithms have been parallelized to run on IBM's shared-nothing multiprocessor SP2. They have been working with parallel implementation of the mining of association rules (APRIORI[6] algorithm) and the SPRINT[25] classification algorithm - an evolution of SLIQ[19] - where multiple processors are able to work together to construct a classification model, namely a decision tree. These algorithms run against data in flat files as well as DB2 family of database products, but databases are accessed in a loosely-couple mode using dynamic SQL.

6.2 DBMiner

DBMiner[15,16] integrates data mining with relational database systems and provides a SQL-like knowledge discovery interface for relational operation-based data generalization, and thus facilitates its integration with the existing commercial relational database systems. It has been designed with emphasis on simplicity and extensibility. Its modules include multiple-level characterization and association, discovery of discriminant rules, classification, and prediction.

6.3 SKICAT

The SKICAT system[10] (Sky Image Cataloging Analysis Tool) was implemented to process images resulting from the Second Palomar Observatory Sky Survey (POSS-II). It uses a variety of classification supervised learning techniques to automating the reduction and analysis of a large astronomical data set, integrating methods for image processing, data classification and database management. Thus, the purpose of SKICAT is to enable and maximize the extraction

of meaningful information from such a large database in an efficient and timely manner. Some learning algorithms (GID3*, O-BTree, and RULER) are used to produce decision trees and classification rules from training data consisting of astronomer-classified sky objects.

6.4 Other Works

Freitas and Lavington[12,13] have presented a series of primitives for rule induction (RI), after analyzing many KDD algorithms of the RI paradigm. Core operations of their candidate rule evaluation procedure are depicted considering performance characteristics and parallelization. Primitives can be translated to usual GROUP BY / ORDER BY clauses in SQL statements, which often provide a potential degree of parallelism in parallel DBMSs.

Holshemeir et al[17] implemented a two-level architecture for constructing decision trees. A DBMS server (Monet Database Server) was used in the experiments, and Quinlan's ID3 algorithm was used as a baseline. Results are presented using a case study with 100k cases (2.6Mb compressed). In parallel execution, a 25k relation was used with 4 nodes.

7 Conclusions

Data mining and its application on large databases have been extensively studied due to the increasing difficulty of analyzing large volumes of data using only OLAP tools. This difficulty pointed out the need of an automated process to discover interesting and hidden patterns in real-world data sets. The ability to handle large amounts of information has been a major concern in many recent data mining applications. Parallel processing comes to play an important role in this context, once only parallel machines can provide sufficient computational power, memory and disk I/O.

We described some important data mining techniques, presenting brief descriptions about them and showing how each one can contribute to the pattern discovery process. Furthermore, we presented several advantages of implementing a data mining method using a DBMS instead of conventional flat files. Our practical work exploited many specific characteristics of DBMSs, providing a tightly-coupled integration of a data mining technique with a parallel database server using a complex application. We have exercised adverse situations

such as large number of attributes, discrete and continuous attributes with many distinct values, observing problems and solutions during the whole process.

Experimental results have shown performance bottlenecks when using a DBMS when compared to flat files, due to the nature of current SQL offered by most commercial databases. However, if we assume that data mining of large databases is going to become in the future a routine task and DBMS vendors are going to implement new features that could help the data mining process, then database servers will play an important role in this context. It is possible that changes to current DBMSs and SQL language could enable data mining operations to be performed more efficiently. Besides, through emerging object-relational technology, a potential area must be exploited. Other interfaces, such as those for integrating with indexing and optimization mechanisms will be available in a near future, which can offer a means of interfering in the parallel optimization process.

References

[1] Agrawal, R., Ghosh, S., Imelinski, T., Iyer, B., & Swami, A., An Interval classifier for database mining applications, *Proc. of VLDB Conference*, Vancouver, Canada, pp. 560-573, 1992.

[2] Agrawal, R., Imielinski, T., & Swami, A., Mining association rules between sets of items in large databases, *Proc. of Int. Conf. ACM SIGMOD*, Washington D. C. pp. 207-216, 1993.

[3] Agrawal, R., Gehrke, J., Gunopulos, & D., Raghavan, P., Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, Seattle, Washington, 1998.

[4] Agrawal, R., Metha, M., Shafer, J., & Srikant, R., The Quest Data Mining System, *Proc. of the 2nd Int. Conf. on Knowledge Discovery in Databases and Data Mining*, Portland, Oregon, 1996.

[5] Agrawal, R., & Shim, K., Developing Tightly-Coupled Data Mining Applications on a Relational Database System, *Proc. of 2nd Int. Conf. on Knowledge Discovery in Databases and Data Mining*, Portland, Oregon, 1996.

[6] Agrawal, R., & Srikant, R., Fast Algorithms for mining association rules, *Proc. of the 20th VLDB Int. Conf.*, Santiago, Chile, 1994.

[7] Bigus, J. P., *Data Mining with NeuralNetworks*, McGraw-Hill, 1996.

[8] Breiman, L., Friedman, J., Olshen, R., & Stone, C., *Classification and Regression Trees*, Wadsworth International Group, 1984.

[9] Chen, M. S., Han, J., & Yu, P. S., Data Mining: An Overview from Database Perspective, *IEEE Trans. on Knowledge and Data Engineering*, Vol. 8, No. 6, pp. 866-883, 1996

[10] Fayyad, U, Djorgovski, S., & Weir, N.. Automating the Analysis and Cataloging of Sky Surveys. *In Advances in Knowledge Discovery and Data Mining*, pp. 471-493, AAAI Press, 1996.

[11] Fayyad, U., Piatesky-Shapriro, G., & Smyth, P., From Data Mining to Knowledge Discovery: An Overview, *In Advances in Knowledge Discovery and Data Mining*, pp. 1-34, AAAI Press, 1996.

[12] Freitas, A., *Generic, Set-oriented Primitives to Support Data-parallel Knowledge Discovery in Relational Database Systems*, Phd diss., 1997, http://cswww.essex.ac.uk/SystemsArchitecture/DataMining /alex/thesis.html.

[13] Freitas, A., & Lavington, S. H., *Mining Very Large Databases With Parallel Processing*, Kluwer Academic Publishers, 1998.

[14] Hallmark, G., Oracle Parallel Warehouse Server, *Proc. of ICDE*, pp. 314-320, 1997.

[15] Han, J., Fu, Y., Koperski, K., Melli, G., Wang, W., & Zaane, O., Knowledge Mining in Databases: An Integration of Machine Learning Methodologies with Database Technologies, *Canadian AI Magazine*, 1995.

[16] Han, J., Fu, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., Li, D., Lu, Y., Rajan, A., Stefanovic, N., Xia, B., & Zaiane, O., DBMiner: A System for Mining Knowledge in Large Relational Databases. *Proc. Int. Conf. on KDD, Portland*, Oregon, 1996.

[17] Holsheimer, M., Kersten, & M. L., Siebes, A., Data Surveyor: Searching the Nuggets in Parallel. *In Advances in Knowledge Discovery and Data Mining*, pp. 447-467, AAAI Press, 1996.

[18] Kufrin, R., Decision Trees on Parallel Processors, *Proc. of the IJCAI Workshop on Parallel Processing for Artificial Intelligence*, pp. 87-95, 1995.

[19] Metha, M., Agrawal, R., & Rissanen, J., SLIQ: A Fast Scalable Classifier for Data Mining, *Proc. of the 5th Int'l Conference on Extending Database Technology (EDBT)*, Avignon, France, 1996.

[20] Metha, M., & DeWitt, D.J., Data Placement in shared-nothing parallel database systems. *VLDB Journal*, Springer-Verlag 1997.

[21] Mitchell, T. M., *Machine Learning*. McGraw-Hill, 1997.

[22] Navathe, S. B., & Ra, M., Vertical Partitioning for Database Design: A Graphical Algorithm. *Proc. of SIGMOD Int. Conf.*, pp. 440-450, 1989.

[23] Oracle Corporation. *Oracle Parallel Server Concepts & Administration Rel. 7.3.* Oracle Technical Manual.

[24] Quinlan, J., *C4.5: Programs for Machine Learning.* Morgan Kaufman, 1993.

[25] Shafer, J., Agrawal, R., & Metha, M., SPRINT: A Scalable Parallel Classifier for Data Mining, *Proc. of the 22th Int. Conf. on VLDB*, Mumbi, India, 1996.

[26] Sousa, M. S., Mattoso, M.L.Q., Ebecken, & N.F.F., Data Mining: A Tightly-Coupled Implementation using a Parallel Database Server. *Proc. Int. Conf. on DEXA Workshop Parallel Databases: innovative applications and new architecture*, IEEE CS, Viena, Austria, 1998.

[27] Sousa, M. S., Mattoso, M.L.Q., Ebecken, & N.F.F., Data Mining on Parallel Database Systems. *Proc. Int. Conf. on PDPTA: Special Session on Parallel Data Warehousing*, CSREA Press, Las Vegas, 1998.

[28] Srikant, R., & Agrawal, R., Mining Sequential Patterns: Generalizations and Performance Improvements, *Proc. of the 5th Int. Conf. on Extending Database Technology (EDBT)*, Avignon, France, 1996.